Complete the following function. This function implements the forward path of a basic LSTM for a single time step.

```
def lstm_cell_forward_path(x, h_prev, c_prev, Wf, Wi, Wc, Wo, bf, bi,
bc, bo):
"""
Args:
x: Input vector at the current time step.
h_prev: Hidden state vector from the previous time step.
c_prev: Cell state vector from the previous time step.
Wf, Wi, Wc, Wo: Weight matrices for the forget gate, input gate, cell
state, and output gate.
bf, bi, bc, bo: Bias vectors for the forget gate, input gate, cell
state, and output gate.

Returns:
h_next: The hidden state vector for the current time step.
c_next: The cell state vector for the current time step.
"""
# Your code here
```

Complete the following function. This function implements the forward path for the self-attention mechanism for a single head Transformer.

```
import numpy as np
def transformer_self_attention(embeddings, W_q, W_k, W_v):
"""
Args:
embeddings: Input tensor of shape (sequence_length, embedding_dim).
W_q: Weight matrix for queries.
W_k: Weight matrix for keys.
W_v: Weight matrix for values.

Returns:
attended_embeddings: Output tensor after applying self-attention.
"""
# Your code here
```

Complete the following function. This function implements the forward path for the self-attention mechanism for a multi head Transformer.

```
import numpy as np
def transformer_self_attention(embeddings, W_q, W_k, W_v,
num_of_heads):
"""
Args:
embeddings: Input tensor of shape (sequence_length, embedding_dim).
W_q: Weight matrix for queries.
W_k: Weight matrix for keys.
W_v: Weight matrix for values.
num_of_heads: Number of attention heads.
Returns:
multi_head_attended_embeddings: Output tensor after applying self-
attention.
"""
# Your code here
```

Write a Python function to calculate positional encodings for a given sequence length and embedding dimension.

```
def positional_encoding(sequence_length, embedding_dim):
"""
Args:
sequence_length: Length of the input sequence.
embedding_dim: Dimensionality of the embedding.

Returns:
positional_encodings: Matrix of positional encodings.
"""
# Your code here
```

Complete the following function. This function implements the forward path for the self-attention mechanism for a single head Transformer.

```
import numpy as np
def transformer_layer(inputs, W_q, W_k, W_v, W_pos, W_ffn1, W_ffn2):
"""
Args:
inputs: Input tensor of shape (batch_size, sequence_length,
embedding_dim).
W_q: Weight matrix for queries.
W_k: Weight matrix for .
W_v: Weight matrix for values.

W_ffn1: Weight matrix for the first layer of the feedforward network.
W_ffn2: Weight matrix for the second layer of the feedforward network.

Returns:
outputs: Output tensor after passing through the Transformer layer.
"""
# Your code here
```